**Practical No. 8**

**Aim:** To implement election algorithm (bully algorithm) in Java

**Theory:**

In [distributed computing](https://en.wikipedia.org/wiki/Distributed_computing), the election algorithm (bully algorithm) is a method for dynamically electing a [coordinator](https://en.wikipedia.org/wiki/Distributed_computing#Coordinator_election) or leader from a group of distributed computer processes. The process with the highest process ID number from amongst the non-failed processes is selected as the coordinator.

The algorithm assumes that:

* The system is synchronous.
* Processes may fail at any time, including during execution of the algorithm.
* A process fails by stopping and returns from failure by restarting.
* There is a failure detector which detects failed processes.
* Message delivery between processes is reliable.
* Each process knows its own process id and address, and that of every other process.

**Algorithm**

The algorithm uses the following message types:

* Election Message: Sent to announce election.
* Answer (Alive) Message: Responds to the Election message.
* Coordinator (Victory) Message: Sent by winner of the election to announce victory.

When a process P recovers from failure, or the failure detector indicates that the current coordinator has failed, P performs the following actions:

1. If P has the highest process id, it sends a Victory message to all other processes and becomes the new Coordinator. Otherwise, P broadcasts an Election message to all other processes with higher process IDs than itself.
2. If P receives no Answer after sending an Election message, then it broadcasts a Victory message to all other processes and becomes the Coordinator.
3. If P receives an Answer from a process with a higher ID, it sends no further messages for this election and waits for a Victory message. (If there is no Victory message after a period of time, it restarts the process at the beginning.)
4. If P receives an Election message from another process with a lower ID it sends an Answer message back and starts the election process at the beginning, by sending an Election message to higher-numbered processes.
5. If P receives a Coordinator message, it treats the sender as the coordinator.

**Program:**

import java.io.\*;

import java.util.Scanner;

class Election{

static int n;

static int pro[] = new int[100];

static int sta[] = new int[100];

static int co;

public static void main(String args[])throws IOException

{

System.out.println("Enter the number of process");

Scanner in = new Scanner(System.in);

n = in.nextInt();

int i,j,k,l,m;

for(i=0;i<n;i++)

{

System.out.println("For process "+(i+1)+":");

System.out.println("Status:");

sta[i]=in.nextInt();

System.out.println("Priority");

pro[i] = in.nextInt();

}

System.out.println("Which process will initiate election?");

int ele = in.nextInt();

elect(ele);

System.out.println("Final coordinator is "+co);

}

static void elect(int ele)

{

ele = ele-1;

co = ele+1;

for(int i=0;i<n;i++)

{

if(pro[ele]<pro[i])

{

System.out.println("Election message is sent from "+(ele+1)+" to "+(i+1));

if(sta[i]==1)

elect(i+1);

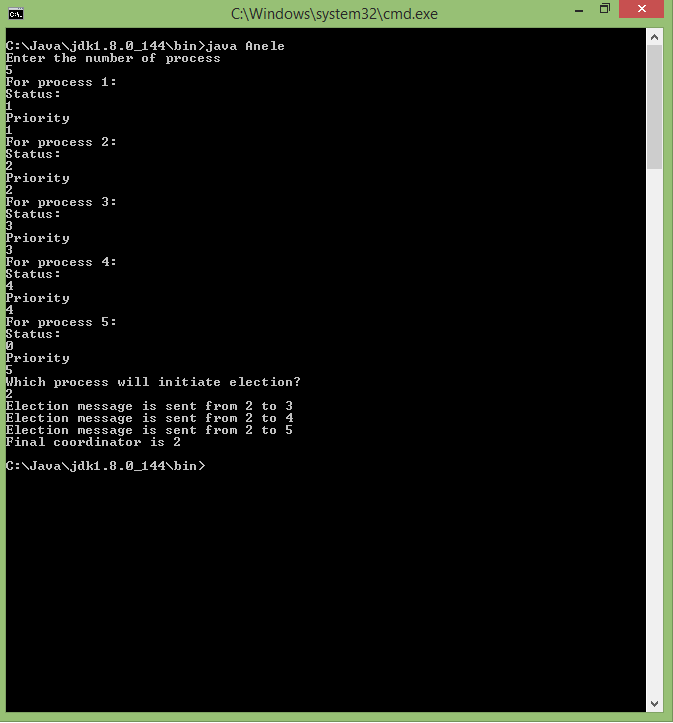
}

}

}

}

**Output:**

****

**Conclusion:** The election algorithm has been implemented and the final coordinator has been determined amongst a group of processes.